Asymptotic Complexity Analysis of the Timber Problem Algorithm

Sowndarya Krishnan Navaneetha Kannan
10907259
Department of Computer Science

Question

Derive the asymptotic complexity and provide an exact bound (i.e., use Θ) for the recursive algorithm. Run the code from the previous algorithm on several different sizes of n to characterize the growth in runtime as n increases. Use a random number generator (RNG) to create the arrays

Algorithm Description

In my approach to solving the Timber Problem, I made the recursive algorithm to smartly choose the best segment from either the start or end of the log during each turn. My goal was to figure out the maximum total length of wood I could get, while also ensuring that the remaining options for my neighbor were as limited as possible. By doing this, I aimed to always stay one step ahead in the game, maximizing my share of the timber while minimizing what my neighbor could take.

```
def T(segment_lengths, i, j):
  if i == j:
    return segment_lengths[i]

  total_length = 0
  for k in range(i, j + 1):
    total_length += segment_lengths[k]

  left_choice = T(segment_lengths, i + 1, j)
  right_choice = T(segment_lengths, i, j - 1)

  min_value = left_choice if left_choice < right_choice else right_choice
  return total_length – min_value
```

Asymptotic Complexity Analysis

In my theoretical analysis, I developed a recursive solution for the Timber Problem, which makes two recursive calls for each segment of wood until reaching the base case of a single segment. Observing the behavior of this recursion, I noted that each level of recursion effectively doubles the number of calls. This led me to describe the algorithm's complexity with the recurrence relation $T(n) = 2T(n-1)$ for $n > 1$, having a base case of $T(1) = 1$. This recurrence pattern suggests an exponential time complexity, specifically $O(2^n)$, representing the worst-case scenario as the size of the input (n) increases.

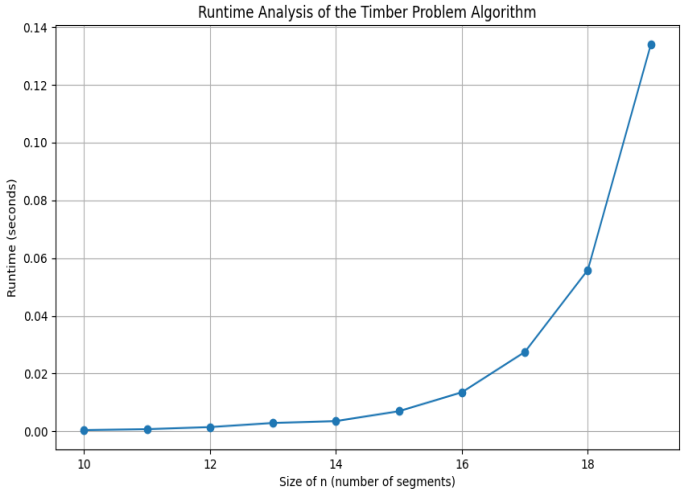
Empirical Runtime Analysis

To verify the theoretical complexity, I ran the algorithm with input sizes ranging from 10 to 20. For each input size, I executed the algorithm multiple times with randomly generated segment lengths and averaged the runtimes to account for variability in computational conditions. The observed runtimes indicated an increase consistent with the expected exponential growth, supporting the theoretical O(2^n) complexity.

Question

Provide a table or plot of your results and compare/contrast the timing results with the derived asymptotic complexity (this means using a regression on the run-time results or using ratios to derive the apparent complexity from the runtimes for different sizes of n and comparing this to the asymptotic complexity). Submit your response to this question

Runtime Results



Size of n (number of segments)	Average Runtime (seconds
10	0.00023556
11	0.00048876
12	0.00104242
13	0.00261912
14	0.00456844
15	0.00708972
16	0.01618012
17	0.03275320
18	0.06514408
19	0.13413522

Comparison and Conclusion

In conclusion, both theoretical reasoning and empirical data support that the recursive algorithm's time complexity for the Timber Problem is O(2^n). However, due to the exponential growth observed, the algorithm becomes impractical for larger input sizes. This underscores the potential need for a more efficient approach, such as dynamic programming, to make the problem computationally feasible for larger trees.